UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/723,967	11/26/2003	Joseph G. Laura	IDF 2584 (4000-16100)	9521

28003 7590 03/08/2007

SPRINT
6391 SPRINT PARKWAY
KSOPHT0101-Z2100
OVERLAND PARK, KS 66251-2100

EXAMINER	
WANG, BEN C	
ART UNIT	PAPER NUMBER
2192	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	03/08/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/723,967	LAURA, JOSEPH G.	
	Examiner	Art Unit	
	Ben C. Wang	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *26 November 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-38* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-38* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *3/15/2004*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-38 are pending in this application and presented for examination.

Specification *Objections*

2. The specification is objected to because the following informalities: "The client 20 may also request the values of variables 116, such as var3 116a" cited in [0045], line 1, should be corrected as "The client 20 may also request the values of variables 110, such as var3 110c"; "monitor module 18 is operable to read the values of the variables 116 from the second" cited in [0045], line 5, should be corrected as "monitor module 18 is operable to read the values of the variables 110 from the second"; "is operable to select the application 20, 22 and to select the variables 110 and 114 to" cited in [0046], line 2, should be corrected as "is operable to select the application 20, 22 and to select the variables 112 and 114 to"; "receive the values of variables 110 and 114 from the client 20 and to display these value" cited in [0046], line 4, should be corrected as "receive the values of variables 112 and 114 from the client 20 and to display these value".

Appropriate correction is required.

Claim Rejections – 35 USC § 102(b)

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102(b) that form the basis for the rejections under this section made in this office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims 1, 3-4, 6-14, and 16-20 are rejected under 35 U.S.C. 102(b) as being anticipated by Zielinski et al. (*A Tool for Monitoring Software-Heterogeneous Distributed Object Applications, 1995, IEEE*) (hereinafter 'Zielinski')

5. **As to claim 1**, Zielinski discloses a system for monitoring an application (Abstract, Lines 8-15; Sec. 1, 1st Para., Lines 6-9; Fig. 1 – A multi-layer architecture including Presentation, Global Monitoring, Interoperability, and Environments), comprising: a first module operable to attach to a memory area used by an application in real-time operation (i.e., Fig. 1, elements of 'Monitored Applications', 'Objects'; Sec. 2, 2nd Para., Lines 1-9 – the environments layer consists of Monitored Applications sub-layer and Local Monitoring sub-layer), the first module further operable to read application values stored in the memory area by the application in real-time operation (i.e., Fig. 1, element of 'Local Monitors', 'LM'; Sec. 2, 2nd Para., Lines 9-12 – the monitored application layer represents original application code, instrumented with notification functions by a set of special preprocessors); a second module in communication with the first module (Sec. 2, 5th Para., Lines 4-6 – interoperability layer can be based on various integration engineers, such as socket) and operable to request the first module to read the application values, the second module further operable to receive the application values from the first module (i.e., Fig. 1, element of 'Global Monitoring', 'GM'; Sec. 2, 4th Para., Lines 1-7); and a third module in communication

with the second module, the third module operable to display the application values (i.e., Fig. 1, element of 'Presentation', 'GUI'; Sec. 2, 4th Para. – cooperates with Graphical User Interface).

6. **As to claim 12**, a method of monitoring operation of an application (Abstract, Lines 8-15; Sec. 1, 1st Para., Lines 6-9; Fig. 1 – A multi-layer architecture including Presentation, Global Monitoring, Interoperability, and Environments), comprising: running an application in a real-time manner (i.e., Fig. 1, elements of 'Monitored Applications', 'Objects'; Sec. 2, 2nd Para., Lines 1-9 – the environments layer consists of Monitored Applications sub-layer and Local Monitoring sub-layer); generating application values stored in a memory area during operation of the application (Fig. 1, element of 'Objects', 'Monitored Applications'); reading the memory area used by the application to obtain the application values (i.e., Fig. 1, element of 'Local Monitors', 'LM'; Sec. 2, 2nd Para., Lines 9-12 – the monitored application layer represents original application code, instrumented with notification functions by a set of special preprocessors); and displaying the application values read from the memory area (i.e., Fig. 1, element of 'Presentation', 'GUI'; Sec. 2, 4th Para. – cooperates with Graphical User Interface).

7. **As to claim 3**, Zielinski discloses the system wherein the first module is further operable to attach to the memory area used by the application to read the application values (i.e., Fig. 1, element of 'Local Monitors', 'LM'; Sec. 2, 2nd Para., Lines 9-12 – the

monitored application layer represents original application code, instrumented with notification functions by a set of special preprocessors).

8. **As to claim 4**, Zielinski discloses the system wherein the application values are further defined as at least one application variable and a value for the application variable (i.e., Fig. 1, elements of 'Monitored Applications', 'Objects'; Sec. 2, 2nd Para., Lines 1-9 – the environments layer consists of Monitored Applications sub-layer and Local Monitoring sub-layer).

9. **As to claim 6**, Zielinski discloses the system wherein the third module is further defined as a graphical user interface (i.e., Fig. 1, element of 'Presentation', 'GUI'; Sec. 2, 4th Para. – cooperates with Graphical User Interface).

10. **As to claim 7**, Zielinski discloses the system wherein the graphical user interface is further operable to receive an input identifying the application values to be read and operable to request the application values identified to the first module (Sec. 2, 4th Para. – cooperates with Graphical User Interface in the process of information visualization, implements the selective monitoring policy, and manages domains of local monitors), via the second module (i.e., Fig. 1, element of 'Global Monitoring', 'GM'; Sec. 2, 4th Para., Lines 1-7), and wherein the first module is operable to read the requested application values data from the memory area (i.e., Fig. 1, element of 'Local Monitors', 'LM'; Sec. 2, 2nd Para., Lines 9-12 – the monitored application layer represents original

application code, instrumented with notification functions by a set of special preprocessors) and return the application variables to the graphical user interface, via the second module.

11. **As to claim 8**, Zielinski discloses the system wherein the graphical user interface is further operable to receive an input identifying requested application values to be displayed (Sec. 2, 4th Para. – cooperates with Graphical User Interface in the process of information visualization, implements the selective monitoring policy, and manages domains of local monitors; Sec. 5, 1st Para., 2nd Para. – GUI's main role is visualization of behavior of monitored systems' selected parts).

12. **As to claim 9**, Zielinski discloses the system wherein the first module is further operable as a socket server and wherein the second module is further operable as a socket client such that the first and second modules communicate via a socket connection (Sec. 4.2, 3rd Para. - 'Socket Solution', 4th Para. – a link between proxy and plug is set up with a socket interface, 6th Para. – we have developed general construction of Interoperability Layer components, i.e. plugs, proxies, gateways and socket modules linked up with local monitors).

13. **As to claim 10**, Zielinski discloses the system wherein the first module operable to read application values stored in the memory area by the application while the application is running (i.e., Fig. 1, element of 'Local Monitors', 'LM'; Sec. 2, 2nd Para.,

Lines 9-12 – the monitored application layer represents original application code, instrumented with notification functions by a set of special preprocessors).

14. **As to claim 11**, Zielinski discloses the system wherein first module operable to read application values stored in the memory area by the application without interfering with the operation of the application (i.e., Fig. 1, element of 'Local Monitors', 'LM'; Sec. 2, 2nd Para., Lines 9-12 – the monitored application layer represents original application code, instrumented with notification functions by a set of special preprocessors).

15. **As to claim 13**, Zielinski discloses the method further comprising: requesting, by a client (i.e., Fig. 1, element of 'Presentation', 'GUI'; Sec. 2, 4th Para. – cooperates with Graphical User Interface), application values from a monitor and wherein the monitor reads the memory area to obtain the application values (i.e., Fig. 1, element of 'Local Monitors', 'LM'; Sec. 2, 2nd Para., Lines 9-12 – the monitored application layer represents original application code, instrumented with notification functions by a set of special preprocessors); and communicating the application variables from the monitor to the client (Sec. 2, 4th Para.).

16. **As to claim 14**, Zielinski discloses the method further comprising: requesting application values (Sec. 3, 3rd Para. – information selecting, filtering etc., according to the chosen management policy); running a plurality of application in a real-time manner (i.e., Fig. 1, elements of 'Monitored Applications', 'Objects'; Sec. 2, 2nd Para., Lines 1-9

– the environments layer consists of Monitored Applications sub-layer and Local Monitoring sub-layer); generating application values stored in one or more memory areas during operation of the plurality of applications (i.e., Fig. 1, elements of 'Monitored Applications', 'Objects'; Sec. 2, 2nd Para., Lines 1-9 – the environments layer consists of Monitored Applications sub-layer and Local Monitoring sub-layer); reading the one or more memory areas used by the plurality of applications to obtain the application values (i.e., Fig. 1, element of 'Local Monitors', 'LM'; Sec. 2, 2nd Para., Lines 9-12 – the monitored application layer represents original application code, instrumented with notification functions by a set of special preprocessors); and displaying the requested application values (i.e., Fig. 1, element of 'Presentation', 'GUI'; Sec. 2, 4th Para. – cooperates with Graphical User Interface).

17. **As to claim 16**, Zielinski discloses the method further comprising providing memory manager (i.e., Fig. 1, element of 'Global Monitoring', 'GM'; Sec. 2, 4th Para., Lines 1-7) and wherein the monitor (i.e., Fig. 1, element of 'Local Monitors', 'LM'; Sec. 2, 2nd Para., Lines 9-12 – the monitored application layer represents original application code, instrumented with notification functions by a set of special preprocessors) registers with the memory manager to obtain a location of the memory area used by the application to store the application values (Fig. 5 – horizontal selection in a tree; Fig. 6 – vertical selection in a tree; Sec. 5.3, 1st Para.).

18. **As to claim 17**, Zielinski discloses the method further comprising: generating new application values by the application stored in the memory area (i.e., Fig. 1, elements of 'Monitored Applications', 'Objects'; Sec. 2, 2nd Para., Lines 1-9 – the environments layer consists of Monitored Applications sub-layer and Local Monitoring sub-layer), at least one of the new application values defined as a new value for a variable of the application (Fig. 5 – horizontal selection in a tree; Fig. 6 – vertical selection in a tree; Sec. 5.3, 1st Para.); requesting, by the client (i.e., Fig. 1, element of 'Presentation', 'GUI'; Sec. 2, 4th Para. – cooperates with Graphical User Interface), that the monitor re-read the application values stored in the memory area; re-reading, by the monitor, the memory area to obtain the new application values (i.e., Fig. 1, element of 'Local Monitors', 'LM'; Sec. 2, 2nd Para., Lines 9-12 – the monitored application layer represents original application code, instrumented with notification functions by a set of special preprocessors).

19. **As to claim 18**, Zielinski discloses the method wherein the monitor reads the application values while the application is running (i.e., Fig. 1, element of 'Local Monitors', 'LM'; Sec. 2, 2nd Para., Lines 9-12 – the monitored application layer represents original application code, instrumented with notification functions by a set of special preprocessors).

20. **As to claim 19**, Zielinski discloses the method wherein the monitor is operable as a socket server and wherein the client is operable as a socket client such that the

communication between the monitor and client is via a socket connection (Sec. 4.2, 3rd Para. - 'Socket Solution', 4th Para. – a link between proxy and plug is set up with a socket interface, 6th Para. – we have developed general construction of Interoperability Layer components, i.e. plugs, proxies, gateways and socket modules linked up with local monitors).

21. **As to claim 20**, Zielinski discloses the method wherein the application values are further defined as a variable of the application and a value of the variable (Sec. 3, 3rd Para.).

Claim Rejections – 35 USC § 102(e)

22. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102(e) that form the basis for the rejections under this section made in this office action:

> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

23. Claims 21-22, 24, 27-28, and 30-31 are rejected under 35 U.S.C. 102(e) as being anticipated by Bates et al. (Pat. No. US 7,086,033 B2) (hereinafter 'Bates')

24. **As to claim 21**, Bates discloses a system for non-intrusively monitoring variables during operation of an application, comprising: a compile listing having an address map with an offset for at least one variable of an application (Col. 1, Lines 33-36; Col. 3, Lines 35-41, 50-51, 53, 54); and a module operable to read the compile listing and obtain the offset of the at least one variable of the application, the module further

operable to attach to an address space where the application is operating to obtain a value for the variable using the offset (Fig. 2; Col. 3, Line 66 through Col. 4, Line 13; Col. 4, Line 44 through Col. 5, Line 4).

25. **As to claim 22**, Bates discloses the system wherein the module is further operable to read the compile listing and convert the variable to the offset (Col. 1, Lines 33-36; Col. 3, Lines 35-41, 50-51, 53, 54; Fig. 2; Col. 3, Line 66 through Col. 4, Line 13; Col. 4, Line 44 through Col. 5, Line 4).

26. **As to claim 23**, Bates discloses the system wherein the module is further operable to search the compile listing (Col. 3, Lines 35-41, 50-51, 53-54) and display a plurality of variables of the application (Fig. 3, element 34 – to display program variable used within program construct; Col. 4, Lines 44-53) for selection by a user.

27. **As to claim 24**, Bates discloses the system wherein the module is responsive to selection by the user of one of the plurality of variables to obtain the value for the selected one of the plurality of variables using the offset to locate the value of the variable in the address space (Col. 1, Lines 33-36; Col. 3, Lines 35-41, 50-51, 53, 54; Fig. 2; Col. 3, Line 66 through Col. 4, Line 13; Col. 4, Line 44 through Col. 5, Line 4).

28. **As to claim 28**, Bates discloses the system wherein the monitor is further operable, using the compile listing (Col. 3, Lines 35-41, 50-51, 53-54), to query the

address map for one or more of the variables of the application (Col. 1, Lines 33-36; Col. 3, Lines 35-41, 50-51, 53, 54).

29. **As to claim 30**, Bates discloses the system wherein the module is further operable to attach to the memory space where the application is operating and overwrite the value for the variable using the offset (Fig. 2; Col. 3, Line 66 through Col. 4, Line 13; Col. 4, Line 44 through Col. 5, Line 4).

30. **As to claim 31**, Bates discloses the system wherein the module comprises: a reader component operable to read the compile listing (Col. 3, Lines 35-41, 50-51, 53-54) and further operable to convert the at least one variable of the application to the offset (Col. 1, Lines 33-36; Col. 3, Lines 35-41, 50-51, 53, 54); and a search component receiving the offset of the at least one variable from the reader component, the search component operable to attach to the application and further operable to locate the value of the variable using the offset (Col. 1, Lines 33-36; Col. 3, Lines 35-41, 50-51, 53, 54).

Claim Rejections – 35 USC § 103(a)

31. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

32. Claims 25-26, 29, and 32-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bates in view of Zielinski

33. **As to claim 25**, Bates does not disclose the system wherein the module is further operable to display the selected one of the plurality of variables.

However, in an analogous art of a tool for monitoring software – heterogeneous distributed object applications, Zielinski discloses the system wherein the module is further operable to display the selected one of the plurality of variables (Fig. 1, element of 'Presentation', 'GUI'; Sec. 3, 3rd Para.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Bates and the teachings of Zielinski to further provide the system wherein the module is further operable to display the selected one of the plurality of variables in Bates system.

The motivation is that it would enhance the Bates system by taking, advancing and/or incorporating the Zielinski system which is expandable, allowing to add new monitored environments, and employs various management mechanisms to cope with gathered information size and complexity for visualization of distributed applications, built of interoperating heterogeneous components as once suggested by Zielinski (i.e., Abstract, Lines 8-15).

34. **As to claim 26**, Bates does not disclose the system wherein the address space is further defined as a memory space and wherein the module attaches, using a socket layer, to the memory space used by the application

However, in an analogous art of a tool for monitoring software – heterogeneous distributed object applications, Zielinski discloses the system wherein the address space is further defined as a memory space and wherein the module attaches, using a socket layer (Sec. 2, 5th Para., Lines 4-6 – interoperability layer can be based on various integration engines, such as socket), to the memory space used by the application (Fig. 1, element of 'Monitored Applications', 'Objects').

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Bates and the teachings of Zielinski to further provide the system wherein the address space is further defined as a memory space and wherein the module attaches, using a socket layer, to the memory space used by the application in Bates system.

The motivation is that it would enhance the Bates system by taking, advancing and/or incorporating the Zielinski system which is expandable, allowing to add new monitored environments, and employs various management mechanisms to cope with gathered information size and complexity for visualization of distributed applications, built of interoperating heterogeneous components as once suggested by Zielinski (i.e., Abstract, Lines 8-15).

35. **As to claim 27**, Bates discloses the system wherein the module attaches, using the offset (Col. 1, Lines 33-36; Col. 3, Lines 35-41, 50-51, 53, 54), to the memory space used by the application via an operating system service (Col. 1, Lines 20-23, 61-63; Fig. 2, element of 25 – Operating System; Col. 3, Line 66 through Col. 4, Line 3; Col. 2, Lines 1-4).

36. **As to claim 29**, Bates does not disclose the system wherein the module is further defined as a subtask of the operating system.

However, in an analogous art of a tool for monitoring software – heterogeneous distributed object applications, Zielinski discloses the system wherein the module is further defined as a subtask of the operating system (i.e. Sec. 2, 2nd Para. – the environments layer consists of an expandable set of popular distributed object-based programming environments; Sec. 4.2, 4th Para. – threads have to be available and used in local monitors and proxy object).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Bates and the teachings of Zielinski to further provide the system wherein the module is further defined as a subtask of the operating system in Bates system.

The motivation is that it would enhance the Bates system by taking, advancing and/or incorporating the Zielinski system which is expandable, allowing to add new monitored environments, and employs various management mechanisms to cope with gathered information size and complexity for visualization of distributed applications,

built of interoperating heterogeneous components as once suggested by Zielinski (i.e., Abstract, Lines 8-15).

37. **As to claim 32**, Bates does not disclose the system further comprising display component operably coupled to the module to receive the value of the variable, the display component operable to display the value.

However, in an analogous art of a tool for monitoring software – heterogeneous distributed object applications, Zielinski discloses the system further comprising display component operably coupled to the module to receive the value of the variable (Sec. 2, 4th Para.), the display component operable to display the value (i.e., Fig. 1, element of 'Presentation', 'GUI'; Sec. 2, 4th Para. – cooperates with Graphical User Interface).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Bates and the teachings of Zielinski to further provide the system further comprising display component operably coupled to the module to receive the value of the variable, the display component operable to display the value in Bates system.

The motivation is that it would enhance the Bates system by taking, advancing and/or incorporating the Zielinski system which is expandable, allowing to add new monitored environments, and employs various management mechanisms to cope with gathered information size and complexity for visualization of distributed applications, built of interoperating heterogeneous components as once suggested by Zielinski (i.e., Abstract, Lines 8-15).

38. **As to claim 33**, Bates does not disclose the system wherein the display component is operable to employ the value to display a heartbeat.

However, in an analogous art of a tool for monitoring software – heterogeneous distributed object applications, Zielinski discloses the system wherein the display component is operable to employ the value to display a heartbeat (Sec. 1, 3rd Para., Lines 1-7 – the system may serve to detect imbalances of the application's configuration and overloads of its component's' interaction, thus allowing appropriate tuning).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Bates and the teachings of Zielinski to further provide the system wherein the display component is operable to employ the value to display a heartbeat in Bates system.

The motivation is that it would enhance the Bates system by taking, advancing and/or incorporating the Zielinski system which is expandable, allowing to add new monitored environments, and employs various management mechanisms to cope with gathered information size and complexity for visualization of distributed applications, built of interoperating heterogeneous components as once suggested by Zielinski (i.e., Abstract, Lines 8-15).

39. **As to claim 34**, Bates does not disclose the system wherein the display component is operable to employ the value to display as a percentage complete.

However, in an analogous art of a tool for monitoring software – heterogeneous distributed object applications, Zielinski discloses the system wherein the display component is operable to employ the value to display as a percentage complete (Sec. 1, 3rd Para., Lines 1-7 – the system may serve to detect imbalances of the application's configuration and overloads of its component's' interaction, thus allowing appropriate tuning).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Bates and the teachings of Zielinski to further provide the system wherein the display component is operable to employ the value to display as a percentage complete in Bates system.

The motivation is that it would enhance the Bates system by taking, advancing and/or incorporating the Zielinski system which is expandable, allowing to add new monitored environments, and employs various management mechanisms to cope with gathered information size and complexity for visualization of distributed applications, built of interoperating heterogeneous components as once suggested by Zielinski (i.e., Abstract, Lines 8-15).

40. Claims 2, 5, 15, and 35-38 are rejected under 35 U.S.C. 103(a) as being unpatentable over Zielinski in view of H. Kashima (*An Approach for Constructing Web Enterprise Systems on Distributed Objects, Jan., 2000, IBM*) (hereinafter 'Kashima')

41. **As to claim 35**, Zielinski does not disclose a system for monitoring COBOL application values, the system comprising: a memory area; a COBOL program operable to generate program values and store the program values in the memory area during real-time operation of the COBOL program; and a COBOL monitor module operable to share the memory area with the COBOL program to read the program values stored in the memory area by the COBOL program.

However, in an analogous art of an approach for constructing web enterprise systems on distributed objects, Kashima discloses a system for monitoring COBOL application values, the system comprising: a memory area; a COBOL program operable to generate program values and store the program values in the memory area during real-time operation of the COBOL program; and a COBOL monitor module operable to share the memory area with the COBOL program to read the program values stored in the memory area by the COBOL program (Sec. 1-4, 3rd Para. – in the case of CORBA, the connectivity with current system is kept high because of its mainframe and COBOL support); Sec. 2, 2nd Para. – the CORBA specification defines IDL, mapping to languages such as COBOL; Sec. 2-1, sub-sec. of 'Language Support' – the CORBA specification specifies the language mapping for COBOL; COBOL support is very important for the products that support mainframe systems).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Zielinski and the teachings of Kashima to further provide a system for monitoring COBOL application values, the system comprising: a memory area; a COBOL program operable to generate program

values and store the program values in the memory area during real-time operation of the COBOL program; and a COBOL monitor module operable to share the memory area with the COBOL program to read the program values stored in the memory area by the COBOL program in Zielinski system.

The motivation is that it would enhance the Zielinski system by taking, advancing and/or incorporating the shared data which will enhance the expandability as a central repository, CORBA (common Object Request Broker Architecture) technologies, and COBOL support which is very important for the products that support mainframe systems as once suggested by Kashima (i.e., Abstract, 2nd Para.; Sec. 2, 2nd Para; Sec. 3-5, sub-sec. of 'shared data'; Sec. 2-1, sub-sec. of 'Language Support').

42. **As to claim 2**, Zielinski does not disclose the system wherein the memory area is further defined as a shared memory of the application.

However, in an analogous art of an approach for constructing web enterprise systems on distributed objects, Kashima discloses the system wherein the memory area is further defined as a shared memory of the application (Sec. 3-5, sub-sec. of Shared Data – this will enhance the expandability if it is centrally provided as in the concept of a repository, because any number of applications can access it).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Zielinski and the teachings of Kashima to further provide the system wherein the memory area is further defined as a shared memory of the application in Zielinski system.

The motivation is that it would enhance the Zielinski system by taking, advancing and/or incorporating the shared data which will enhance the expandability as a central repository, CORBA (common Object Request Broker Architecture) technologies, and COBOL support which is very important for the products that support mainframe systems as once suggested by Kashima (i.e., Abstract, 2nd Para.; Sec. 2, 2nd Para; Sec. 3-5, sub-sec. of 'shared data'; Sec. 2-1, sub-sec. of 'Language Support').

43. **As to claim 5**, Zielinski does not disclose the system wherein the first module is further operable to communicate the application values to the second module in hypertext markup language format.

However, in an analogous art of an approach for constructing web enterprise systems on distributed objects, Kashima discloses the system wherein the first module is further operable to communicate the application values to the second module in hypertext markup language format (i.e., Abstract, 1st Para.; Sec. 1-2, 1st Para.; Sec. 1-3, 5th Para., Lines 1-3).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Zielinski and the teachings of Kashima to further provide the system wherein the first module is further operable to communicate the application values to the second module in hypertext markup language format in Zielinski system.

The motivation is that it would enhance the Zielinski system by taking, advancing and/or incorporating the shared data which will enhance the expandability as a central

repository, CORBA (common Object Request Broker Architecture) technologies, and COBOL support which is very important for the products that support mainframe systems as once suggested by Kashima (i.e., Abstract, 2nd Para.; Sec. 2, 2nd Para; Sec. 3-5, sub-sec. of 'shared data'; Sec. 2-1, sub-sec. of 'Language Support').

44. **As to claim 15**, Zielinski does not disclose the method wherein the memory area is further defined as a block of shared memory and wherein the monitor reads the at least some of the application variables stored in the block of shared memory.

However, in an analogous art of an approach for constructing web enterprise systems on distributed objects, Kashima discloses the method wherein the memory area is further defined as a block of shared memory and wherein the monitor reads the at least some of the application variables stored in the block of shared memory (Sec. 3-5, sub-sec. of Shared Data – this will enhance the expandability if it is centrally provided as in the concept of a repository, because any number of applications can access it).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Zielinski and the teachings of Kashima to further provide the method wherein the memory area is further defined as a block of shared memory and wherein the monitor reads the at least some of the application variables stored in the block of shared memory in Zielinski system.

The motivation is that it would enhance the Zielinski system by taking, advancing and/or incorporating the shared data which will enhance the expandability as a central repository, CORBA (common Object Request Broker Architecture) technologies, and

COBOL support which is very important for the products that support mainframe systems as once suggested by Kashima (i.e., Abstract, 2nd Para.; Sec. 2, 2nd Para; Sec. 3-5, sub-sec. of 'shared data'; Sec. 2-1, sub-sec. of 'Language Support').

45. **As to claim 36**, Zielinski does not disclose the system further comprising: a second COBOL program operable to generate second program values and store the program values in the memory area during real-time operation of the second COBOL program, and wherein the COBOL monitor module is further operable to read the second program values stored in the memory area by the second COBOL program.

However, in an analogous art of an approach for constructing web enterprise systems on distributed objects, Kashima discloses the system further comprising: a second COBOL program operable to generate second program values and store the program values in the memory area during real-time operation of the second COBOL program, and wherein the COBOL monitor module is further operable to read the second program values stored in the memory area by the second COBOL program (Sec. 1-4, 3rd Para. – in the case of CORBA, the connectivity with current system is kept high because of its mainframe and COBOL support); Sec. 2, 2nd Para. – the CORBA specification defines IDL, mapping to languages such as COBOL; Sec. 2-1, sub-sec. of 'Language Support' – the CORBA specification specifies the language mapping for COBOL; COBOL support is very important for the products that support mainframe systems).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Zielinski and the teachings of Kashima to further provide the system further comprising: a second COBOL program operable to generate second program values and store the program values in the memory area during real-time operation of the second COBOL program, and wherein the COBOL monitor module is further operable to read the second program values stored in the memory area by the second COBOL program in Zielinski system.

The motivation is that it would enhance the Zielinski system by taking, advancing and/or incorporating the shared data which will enhance the expandability as a central repository, CORBA (common Object Request Broker Architecture) technologies, and COBOL support which is very important for the products that support mainframe systems as once suggested by Kashima (i.e., Abstract, 2nd Para.; Sec. 2, 2nd Para; Sec. 3-5, sub-sec. of 'shared data'; Sec. 2-1, sub-sec. of 'Language Support').

46. **As to claim 37**, Zielinski does not disclose the system further comprising: a second memory area; and a second COBOL program operable to generate second program values and store the program values in the second memory area during real-time operation of the second COBOL program, and wherein the COBOL monitor module is further operable to read the second program values stored in the second memory area by the second COBOL program.

However, in an analogous art of an approach for constructing web enterprise systems on distributed objects, Kashima discloses the system further comprising: a

second memory area; and a second COBOL program operable to generate second program values and store the program values in the second memory area during real-time operation of the second COBOL program, and wherein the COBOL monitor module is further operable to read the second program values stored in the second memory area by the second COBOL program (Sec. 1-4, 3rd Para. – in the case of CORBA, the connectivity with current system is kept high because of its mainframe and COBOL support); Sec. 2, 2nd Para. – the CORBA specification defines IDL, mapping to languages such as COBOL; Sec. 2-1, sub-sec. of 'Language Support' – the CORBA specification specifies the language mapping for COBOL; COBOL support is very important for the products that support mainframe systems).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Zielinski and the teachings of Kashima to further provide the system further comprising: a second memory area; and a second COBOL program operable to generate second program values and store the program values in the second memory area during real-time operation of the second COBOL program, and wherein the COBOL monitor module is further operable to read the second program values stored in the second memory area by the second COBOL program in Zielinski system.

The motivation is that it would enhance the Zielinski system by taking, advancing and/or incorporating the shared data which will enhance the expandability as a central repository, CORBA (common Object Request Broker Architecture) technologies, and COBOL support which is very important for the products that support mainframe

systems as once suggested by Kashima (i.e., Abstract, 2nd Para.; Sec. 2, 2nd Para; Sec. 3-5, sub-sec. of 'shared data'; Sec. 2-1, sub-sec. of 'Language Support').

47. **As to claim 38**, Zielinski does not disclose the system further comprising: a user interface operable to monitor and display the application values; and a client application in communication with the user interface and the COBOL monitor module, the client application operable to request the program variables of the COBOL program from the COBOL monitor module and provide the program variables to the user interface for display via the user interface responsive to a request from the user interface.

However, in an analogous art of an approach for constructing web enterprise systems on distributed objects, Kashima discloses the system further comprising: a user interface operable to monitor and display the application values; and a client application in communication with the user interface and the COBOL monitor module, the client application operable to request the program variables of the COBOL program from the COBOL monitor module and provide the program variables to the user interface for display via the user interface responsive to a request from the user interface (Sec. 1-4, 3rd Para. – in the case of CORBA, the connectivity with current system is kept high because of its mainframe and COBOL support); Sec. 2, 2nd Para. – the CORBA specification defines IDL, mapping to languages such as COBOL; Sec. 2-1, sub-sec. of 'Language Support' – the CORBA specification specifies the language mapping for COBOL; COBOL support is very important for the products that support mainframe systems).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Zielinski and the teachings of Kashima to further provide the system further comprising: a user interface operable to monitor and display the application values; and a client application in communication with the user interface and the COBOL monitor module, the client application operable to request the program variables of the COBOL program from the COBOL monitor module and provide the program variables to the user interface for display via the user interface responsive to a request from the user interface in Zielinski system.

The motivation is that it would enhance the Zielinski system by taking, advancing and/or incorporating the shared data which will enhance the expandability as a central repository, CORBA (common Object Request Broker Architecture) technologies, and COBOL support which is very important for the products that support mainframe systems as once suggested by Kashima (i.e., Abstract, 2nd Para.; Sec. 2, 2nd Para; Sec. 3-5, sub-sec. of 'shared data'; Sec. 2-1, sub-sec. of 'Language Support').

Conclusion

48. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Bates et al., *Descriptive Variables While Debugging* (Pub. No. US 2003/0221186 A1)

- K. J. Hines, *System and Method for Debugging Distributed Software Environments* (Pub. No. US 2002/0174415 A1)
- Bates et al., *Displaying Variable Usage While Debugging* (Pat. No. US 6,961,924 B2)
- G. C. Hunt, *Dynamic Classification of Sections of Software* (Pat. No. US 6,957,422 B2)

49. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

BCW

February 18, 2007